ANNO 1-N. 4 L. 3.900

PC 128-PC 128S

olivetti PRODEST USER

LA PRIMA E UNICA RIVISTA INDIPENDENTE PER GLI UTENTI PC 128 E PC 128 S

Numero 4 - Aprile-Maggio '87

Introduzione al BASIC del PC 128S

Colorcalc: foglio elettronico

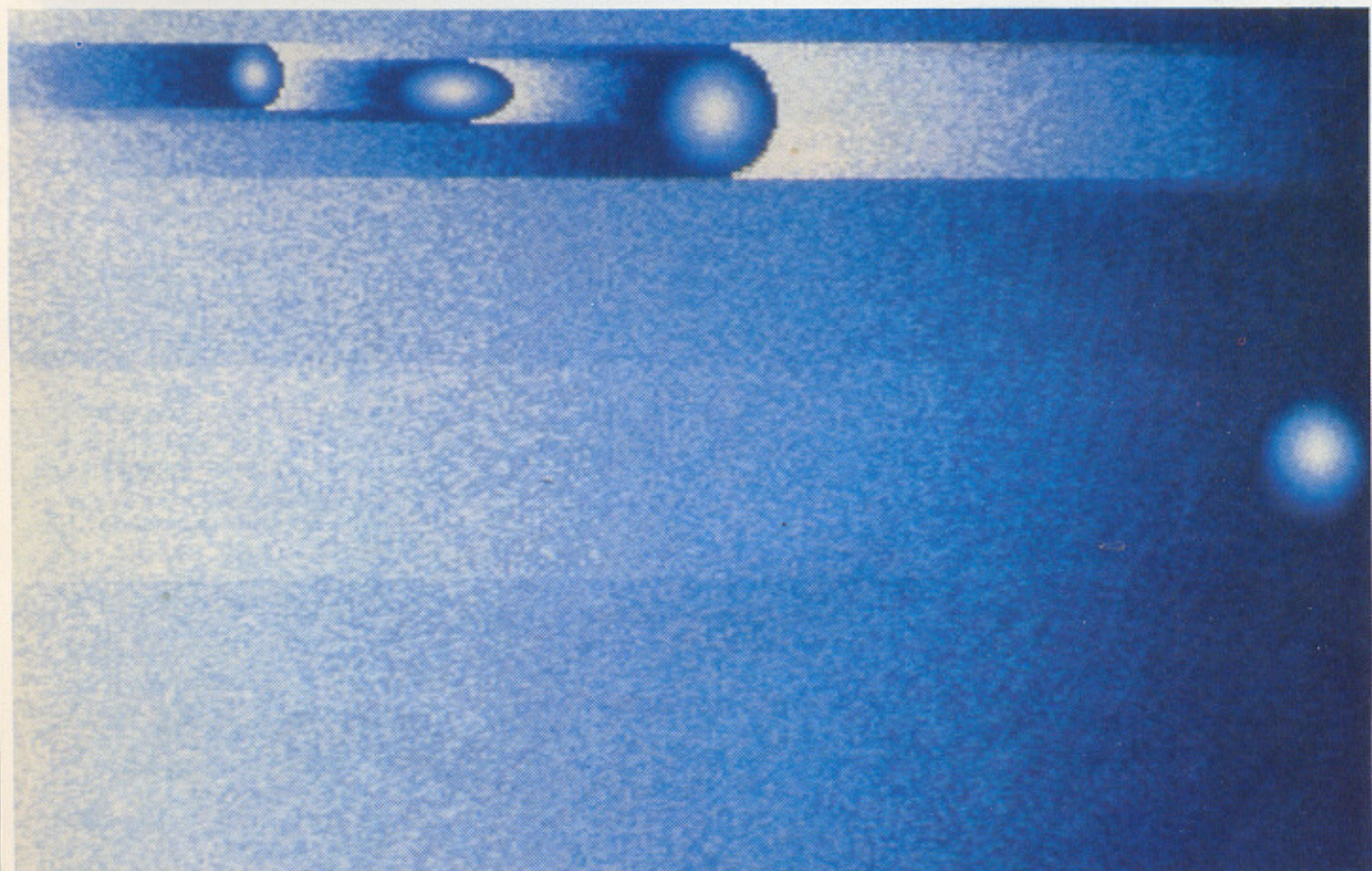
Sistema musicale

Intervista con: Gianna Nannini

Il PC 128S e l'emulazione MS-DOS

Spedizione in abbonamento postale Gruppo III/70

INTRODUZIONE AL BASIC DEL PC 128S

Il Basic di cui è dotato il PC 128S, è il BASIC IV. Esso è la necessaria evoluzione dei vari BASIC I, II e III e perciò vedremo ora di compararne alcuni comandi e di scoprirne le facilitazioni nella gestione delle più diverse procedure, anche le più complesse, tramite l'uso di pochi ma appropriati comandi. Questa sezione della rivista sarà particolarmente utile a chi si avvicina per la prima volta al mondo dei computer e ai suoi linguaggi, ma potrà essere di utile apporto anche a coloro che, più esperti, vogliano conoscere a fondo le peculiarità del BASIC del PC 128S. In questa prima parte vedremo di esaminare questo linguaggio accennando alle sue caratteristiche.

Il comando ON

Con questo comando è possibile richiamare delle procedure ed anche delle subroutines.

Per esempio:

120 ON in% − 129 PROCleft, PROCright, PROCdown, PROCup, ELSE PROCerr.

La chiamata PROC ha esattamente la stessa sintassi delle normali procedure di chiamata, può avere cioè parametri separati da virgole fra parentesi tonde:

1240 ON menu% PROCadd (name$,ag%), PROCdel (name$), PROCshow (name$).

La costruzione LIST IF

Il comando LIST è stato esteso e contempla una funzione di "cross-reference". Il comando LIST del tipo LIST, oppure LIST 200, 1000, può essere seguito dalla parola chiave IF, la quale può essere a sua volta seguita da una stringa. Solo le linee di programma contenenti la stringa verranno listate. I caratteri che seguono l'IF sono scritti così come vengono usati, in tal modo possono essere create le parole chiave.
Per esempio:

LIST IF lista i comandi IF

LIST IF TIME lista le linee contenenti TIME come una funzione

LIST IF name $(lista le linee che usano array name $(

LIST 100, 200 IF var= lista i valori attribuiti a var nelle linee da 100 a 00

Il comando EDIT

Ora è possibile usufruire di potenti agevolazioni del "text editor" per redigere un programma BASIC senza dover fare lo *SPOOL dello stesso, usando il nuovo comando EDIT. EDIT edita a proprio carico l'intero programma, ma può essere usato, con gli stessi parametri del LIST (inclusa la parte IF), per editare solo parte del programma.

Data l'importanza di questo comando, una sua trattazione più ampia verrà fatta in un prossimo numero.

La pseudo variabile TIME$

È riferita al tempo e provvede all'accesso al tempo reale del sistema. Può essere usata come una funzione (ex. PRINT TIME$) o come un'istruzione (ex. TIME$="...").

EXT# =

Nel BASIC II la funzione EXT# = è usata per visualizzare la lunghezza di un file sequenziale aperto. La suddetta facilitazione può essere usata solamente con dei filing system idonei (ADSF, Network).

Color

Nel BASIC IV l'istruzione COLOUR può essere anche scritta come COLOR. In ogni caso sarà egualmente accettata, ma verrà comunque listata come COLOUR.

Estensioni all'assembler

Il microprocessore usato nel PC 128S è il 65C12, una versione CMOS del 6502 con un set di istruzioni arricchito. Per avvantaggiarsi di ciò, l'assembler costruito nel BASIC IV è stato esteso per poter accettare le nuove istruzioni.

I listati dell'assembler sono ora strutturati in una maniera più leggibile, tanto che le label possono avere fino a nove caratteri, incluse le iniziali.

Si possono usare i caratteri minuscoli in ogni parte del programma sorgente, come 1 da &70, x and equs "fred".

L'assembler non va più in stato confusionale a causa del modo d'indirizzamento dell'accumulatore. Per esempio, nelle versioni precedenti, ASL ALFRED veniva letto come ASL A, seguito dal commento LFRED. Ciò non avviene più.

Il BASIC attiva alcuni indirizzi puntati a delle routine di floating poin. Questi possono essere richiamati da programmatori esperti in linguaggio macchina.

Altri cambiamenti minori

Il SAVE può prendere qualsiasi stringa come proprio argomento (ex. SAVE a$+b$], mentre il "!" e "?" possono essere usati come parametri formali (DEF FN ferd(!&70)).

Ora si può usare il codice ASCII 141 nei commenti e nelle stringhe: per esempio per produrre caratteri in doppia altezza nei modi teletext:

100 REM <&8D>Big comment
110 REM <&8D>Big comment

(Prima il RENUMBER e il LIST sarebbero stati confusi da questo codice.)

In addizione, LIST potrà listare commenti che includono codici di teletext colour, come linee colorate nei modi teletext; non avrai più bisogno di chiudere fra virgolette le REM.

Introduzione per i programmatori del microsoft BASIC

Aiuti per la programmazione strutturata.

Il BASIC del PC 128S, dispone di diverse caratteristiche che aiutano il programmatore nella stesura di programmi leggibili e ben strutturati. I comandi in questione sono:

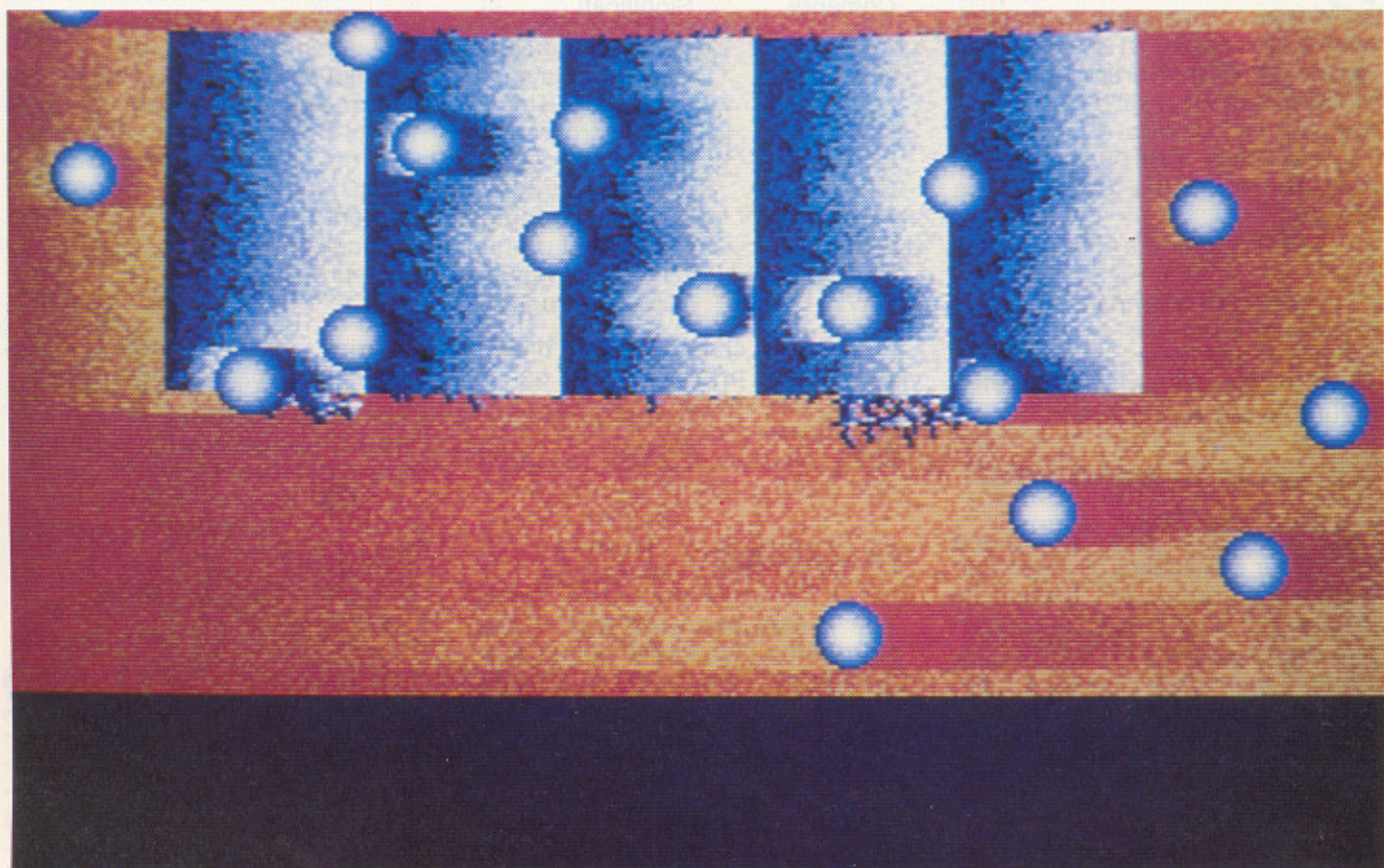
— l'istruzione IF...THEN...ELSE...
— la costruzione REPEAT...UNTIL...
— l'istruzione ON...PROC...
— procedure definibili
— funzioni definibili
— parametri di procedure e di funzioni e variabili locali.

I vantaggi di aiutare la stesura di programmi leggibili sono molteplici:

— possibilità di adoperare nomi di variabili lunghi e significativi
— identificazione automatica di loop, se richiesta
— un potente comando RENUMBER

In addizione a questi, ci sono i soliti comandi BASIC quali GOTO, GOSUB, ON...GOTO/GOSUB e FOR...NEXT.

Accessi al sistema operativo

Il computer dispone di un potente sistema operativo (MOS), che controlla l'hardware della macchina

(monitor, tastiera, convertitori analogico-digitali, stampanti, ecc.). Il MOS inoltre, supporta i filing system, usati per immagazzinare o richiamare informazioni nei floppy disc, cartridge, ecc. Il BASIC del PC 128S dispone di un pieno accesso al MOS e ai filing system per mezzo di istruzioni come PLOT, KEY, ADVAL, OSCLI, ecc. Questi comandi sono molto più facili da usare e da capire dei loro "oscuri" predecessori PEEK e POKE.

Accessi al codice macchina

Sebbene il BASIC IV del PC 128S sia attualmente la più veloce versione di BASIC a otto bit disponibile, ci sono dei casi, come per esempio l'estrazione veloce di array molto vaste, in cui è richiesta la velocità del linguaggio macchina. A tal fine, il BASIC dispone di un potente assembler 65C12, che può essere usato per sviluppare diversi tipi di programmi in tale linguaggio.

L'istruzione CALL può essere usata per chiamare le routine in linguaggio macchina e per passare loro dei parametri BASIC d'ogni tipo.

La funzione USR può essere usata per chiamare le routine del codice macchina inizializzando i registri del 6502 con le variabili del BASIC e riportando il contenuto del registro in uscita.

CALL e USR dispongono di un facile accesso alle routine del sistema operativo, che non sono direttamente supportate dalle funzioni BASIC.

Trattamento dei dati

Il BASIC del PC 128S supporta numeri interi a quattro byte (a differenza dei soliti due byte), e numeri floating point a cinque byte. Le stringhe possono essere assegnate dinamicamente e possono arrivare a 255 caratteri di lunghezza. Le array multi-dimensionali sono utilizzabili come array di singoli byte.

L'accesso alla memoria è possibile grazie agli operatori indiretti "?", "!" e "$". "?" ha degli usi similari al PEEK e POKE (i quali non sono né necessari né supportati dal BASIC del PC 128S). "!" è simile, ma agisce su quattro byte alla volta, invece di un singolo byte. "$" agisce sulle stringhe di carattere (questo è provvisto in addizione alle normali variabili stringa).

Un ricco set di funzioni numeriche e di stringa prevede funzioni logaritmiche e trigonometriche, gestione delle stringhe e funzioni di ricerca.

Formati di stampa

I numeri possono essere rappresentati in tre formati (generale, esponente, fisso), con campi di lar-

ghezza variabili. I numeri interi possono essere stampati in esadecimale e la funzione STR$ può essere usata per convertire numeri in stringhe esadecimali o per dare il formato della stringa.

L'istruzione PRINT ha diversi modi di gestione della pagina schermo:

— ' per stampare una nuova linea

— SPC per stampare un certo numero di spazi

— TAB serve per posizionare l'inizio di una stringa sullo schermo da una data posizione.

Trattamento degli errori

ON ERROR è una facilitazione fornita per individuare gli errori non fatali e per trattarli per mezzo del programma. Il comando è supportato dalle funzioni ERR e ERC, che danno il numero dell'errore e il numero di linea dell'ultimo errore commesso e dalla funzione REPORT, che stampa l'ultimo messaggio d'errore.

Il modo diretto

Il prompt del BASIC è ">", e la sua presenza all'inizio di una linea indica che il BASIC è pronto ad accettare l'input. Il programmatore può digitare dei comandi come AUTO e LIST, i quali saranno immediatamente eseguiti, come PRINT LOG (12), o linee BASIC che devono essere inserite nel programma. Queste ultime dovranno essere precedute da un numero di linea nel campo 0-32767. Le singole linee del programma possono essere cancellate digitando il numero di linea immediatamente seguito da RETURN.

Sotto riportiamo una lista di alcuni comandi con i loro significati. Notare che un comando non può essere eseguito dall'interno di un programma e non può essere preceduto da un altro comando o da un'istruzione. Se seguito da una istruzione, quest'ultima verrà ignorata.

Comandi	Significati
AUTO	Genera automaticamente i numeri di linea
DELETE	Cancella una gamma di numeri di linea da un programma
EDIT	Chiama il system editor per redigere il programma BASIC
LIST	Lista il programma
LISTO	Setta l'opzione indentation del LIST
LOAD	Carica un programma BASIC
NEW	Cancella il programma corrente
OLD	Richiama il programma cancellato da NEW
RENUMBER	Rimette in sequenza i numeri di linea
SAVE	Salva un programma BASIC

I comandi che seguono, a differenza dei primi, possono essere usati anche in modo programma, ma generalmente vengono usati in modo diretto.

CHAIN	Carica ed esegue un programma BASIC
RUN	Esegue un programma BASIC già residente.

Le variabili

In questa parte dell'articolo descriveremo i tipi di dati che possono essere usati nei programmi scritti in BASIC IV del PC 128S e gli operatori e le funzioni previste per trattare gli stessi.

Tipi di variabili in BASIC

Nel BASIC del PC 128S ci sono tre tipi di dati fondamentali: numeri reali, numeri interi e stringhe. Per ognuno c'è un tipo di variabile corrispondente: reale, intera e stringa. Esiste una facilitazione che permette di dichiarare delle array multi-dimensionali di ogni tipo. Ogni tipo di variabile dispone di un set di funzioni e di operatori con i quali operare. Notare che i numeri reali e gli interi sono largamente intercambiabili; il BASIC, quando richiesto, converte automaticamente i valori in valori interi e viceversa.

Numeri reali

Le variabili di tipo reale sono semplicemente degli identificatori. In BASIC un identificatore è una sequenza di uno o più caratteri nel set [A, B.. Y, Z, a, b,... y, z, 0, 1..8,9, —, £]. Il primo carattere non può essere una cifra. Esempi di variabili reali sono:

£amount
numVars
A—Long—Variable—Name
vas 1234—21z
—

Si può vedere come i caratteri speciali — (underline) e £ (pound), agiscono come lettere extra. Una restrizione degli identificatori da non dimenticare è che questi non possono cominciare con tutte le parole riservate. Per esempio, GETADDR è illegale, poiché GET è una parola riservata. Tuttavia le parole riservate possono essere incastonate in un identificatore, come in NAMELIST. Poiché le parole riservate devono essere in maiuscolo, gli identificatori come getaddr e anche list possono essere usati. Le parole riservate, utilizzabili all'inzio di un identificatore, sono quelle che normalmente non sono seguite da qualcosa: CLEAR, CLG, CLS, COUNT, END, ENDPROC, ERL, ERR, FALSE, HIMEN, LOMEN, NEW, OLD, PAGE, PI, POS, REPORT, RETURN, RUN, STOP, TIME, TRUE, VPOS. In questo modo, variabili come COUNTER e POST sono legali e distinte da COUNT e POS.

Le array reali sono semplicemente degli identificatori seguiti da sotto scritture tra parentesi:

counts (char-128)
mat (i%, j%)
numVars (0)

Notare che la array numVars( ) e la semplice variabile numVars

possono essere entrambe nello stesso programma; esse sono entità completamente separate. Una costante reale è una sequenza di cifre con una parte decimale opzionale e una parte esponente opzionale. Esempi sono:

1224
12.34
—0.000001
.123
10293.1234
+0.1E10
−112.32E−4

Il segno del numero, se non dato esplicitamente, è considerato sempre positivo. Il numero <n> dopo la E significa "moltiplicare il numero per 10 alla potenza di n". La grandezza massima di un numero reale, che può essere rappresentata dal BASIC è 1.701411834E+38. Il tentativo di generare un valore più grande di questo produrrà un errore. La grandezza minima di un numero reale (questo è il numero più vicino a 0) è 1,469367939E-39. Il tentativo di generare un numero inferiore a questo non avrà successo.

Numeri interi

Le variabili intere sono degli identificatori seguiti dal segno "%". Esempi sono:

check-sum%
£pennies%
cx%
YCOORD1%

Gli elementi delle array intere sono indicati nello stesso modo delle array reali: facendo seguire alla variabile una lista di sotto-scritti. Esempi sono:

lookup%(i%)
board%(row, col)

Le costanti intere vengono scritte come sequenze tra una e dieci cifre significative, opzionalmente precedute da un segno. Esempio sono:

−99
234134112
−532354
+42

Le costanti intere possono essere scritte anche in esadecimale, facendo precedere alla parte costante una "&". Il numero è una sequenza da una a otto cifre esadecimali significative (0,1..8,9,A,B..E,F). Esempio sono:

&OD
&FF7FF80
−&55AA1
&80000000

Non ci sono spazi fra la "&" e la prima cifra.

Le variabili intere del sistema

Ci sono 27 variabili intere, che sono definite permanentemente e vengono appunto chiamate: Variabili Intere del Sistema. Esse sono A%-Z% e @% %.
@% è usata per controllare il formato di stampa (vedi l'istruzione PRINT). 0% e P% hanno un significato particolare quando si programma in codice macchina. A%, C%, X% e Y% hanno un significato particolare quando si usano i programmi in codice macchina. Le altre variabili intere del sistema sono completamente libere all'uso del programmatore: come anche A%, C%, O%, P%, X% e Y% se non usate in linguaggio macchina.

Il vantaggio principale delle variabili di sistema è che non vengono azzerate da CLEAR, NEW, OLD, RUN, LOAD e CHAIN; pertanto possono essere usate per passare informazioni attraverso i programmi. Un altro vantaggio è che il BASIC permette un'entrata molto veloce delle variabili di sistema, perché conosce esattamente la loro posizione in memoria. È pertanto una buona idea usare queste variabili quando risulta importante la velocità (come all'interno di un loop FOR... NEXT).

Altre variabili possono essere create usando un'assegnazione oppure l'istruzione INPUT e possono essere cancellate usando le istruzioni sopra menzionate.

Numeri interi indiretti

C'è un'altra forma di variabili intere, accessibile per mezzo del prefisso "!". Il formato di questa variabile è !<factor>, dove <factor> è un numero, una variabile o una funzione di chiamata o anche un'espressione fra parentesi. Alcuni tipici numeri interi indiretti sono:

!&70
!ptr%
!words%(i%)

Il valore di un numero intero indiretto, è il valore dei quattro bytes all'indirizzo specificato. In questo modo, !&70 ha un valore determinato dai quattro bytes, complemento di due, immagazzinati alla locazione &70-&73. Analogalmente, !ptr% assume ptr% come indirizzo di un valore a 4 byte alla locazione ptr%-pt%+3 e dà il valore di questo numero intero.

C'è un'estensione alla notazione "!": essa ha la forma <variabile>!<factor>. Questa volta, <variabile>, è una variabile numerica (reale o intera). Il valore ottenuto è un numero intero a 4 byte alla locazione <variabile> + <-factor> — <variabile> + <factor> +3. Da questo si può vedere che <variabile>!<factor> è solamente un altro modo di scrivere !(<variabile><+factor>). Ecco alcuni esempi:

table! 10
vals%! i%
address%!(2*index%)

Stringhe

Le variabili stringa possono supportare stringhe lunghe fino a 255 caratteri. L'estensione minore della stringa è la stringa nulla, che ha una lunghezza pari a zero. L'indirizzo di una variabile stringa, effettivamente l'indirizzo del suo "String Information Bloc" (SIB), è lungo quattro bytes, ed ha il formato:

Address of string text SIB + 0
Bytes allocated to the string SIB + 2
Current lenght of string SIB + 3.

Per ottenere il meglio del BASIC, è meglio sapere esattamente quante sono le stringhe immagazzinate. Questo è particolarmente importante se si stanno scrivendo programmi piuttosto lunghi, con parecchie stringhe, poiché il BASIC del PC 128S non esegue nessun

"garbage collection". Ci sono però delle tecniche di programmazione che permettono un notevole risparmio di memoria.

Quando una variabile stringa viene creata (per esempio grazie ad un'istruzione di assegnazione), il BASIC respinge il suo immagazzinamento. Se la lunghezza iniziale è inferiore agli otto caratteri, l'immagazzinamento (il valore dei bytes allocati) è lo stesso della lunghezza. Se la lunghezza iniziale è di otto o più caratteri, i bytes allocati saranno maggiorati di otto caratteri rispetto all'originale, fino ad un limite massimo di 255 caratteri. Quindi per esempio:

```
a$="123456"
b$="1234567890"
```

verranno usati 6 bytes per a$ (gli stessi della sua lunghezza) e 18 bytes per b$ (8 in più dell'originale). Collocando più caratteri dell'originale lunghezza, il BASIC permette alla stringa di "crescere" prima di doverle trovare una nuova area di immagazzinamento. Questo è importante perché, se la stringa sorpassa la sua collocazione originale, essa viene spostata in un'area più grande che la possa contenere. In tal modo, l'area che essa occupava in precedenza non è più accessibile e non può essere riusata, dando così luogo ad un enorme spreco di memoria.

Consideriamo ciò che accade quando a$ e b$ vengono allungate di due bytes, per esempio attraverso le istruzioni:

```
a$=a$+"AB"
b$=b$+"AB"
```

La stringa a$ occuperà ora 8 bytes. Precedentemente a$ aveva a disposizione uno spazio di memoria pari a 6 bytes. Dopo l'operazione di somma delle stringhe, a$ verrà rilocata in un altro spazio di memoria grande 8+8 bytes, lasciando però inutilizzabile i primi 6 bytes originari.

Quando la b$ è riassegnata, la sua nuova lunghezza è di 10 bytes, ma lo spazio assegnatole dal BASIC era di 18 bytes, cosicché essa non dovrà essere spostata in un altro luogo della memoria e di conseguenza non ci sarà spreco di preziosi bytes.

L'unica eccezione è che, se una variabile stringa è l'ultima variabile dinamica ad essere impostata, essa può crescere fino alla massima lunghezza permessa senza richiedere nuovo spazio. Questo accade perché il BASIC sa che non c'è nulla dopo di essa e che può essere estesa senza che ciò alteri alcun dato. È ovvio che se dobbiamo adoperare una stringa con un ampio campo di variazioni, sarà opportuno inserirla per ultima nelle variabili create dal programma. Bisogna però fare attenzione, perché le variabili LOCAL e i parametri procedura/funzione, possono creare variabili anche se non ancora esistenti.

Riassumendo: sarà utile assegnare subito ad una variabile stringa la massima lunghezza che si presume raggiungerà. Naturalmente subito dopo, si dovrà resettarla come segue:

```
a$=STRING$(30, "*")
a$=" "
```

Usando questo metodo sarà possibile:

— accelerare l'esecuzione del programma
— prevenire gli errori "No room" causati dal continuo spostamento delle stringhe in memoria.

Le costanti stringa sono sequenze di caratteri tra 0 e 255, racchiuse tra virgolette. Il numero massimo dei caratteri in una stringa è dato anche dal numero massimo di caratteri accettati dalla "linea BASIC", cioè 238.

Per poter includere in una stringa le virgolette si dovrà digitarle due volte.

Esempi di costanti stringa sono:

```
"A string constant"
"A string" "with a quote in"
" ": REM stringa nulla
" " " ": REM doppie virgolette
```

IL PC 128S E L'EMULAZIONE MS-DOS

Un utile programma che permette ai possessori di un sistema PC 128S, la manipolazione di file sotto MS-DOS e il passaggio di file, per esempio quelli ottenibili con View, da ADFS a sistemi MS-DOS

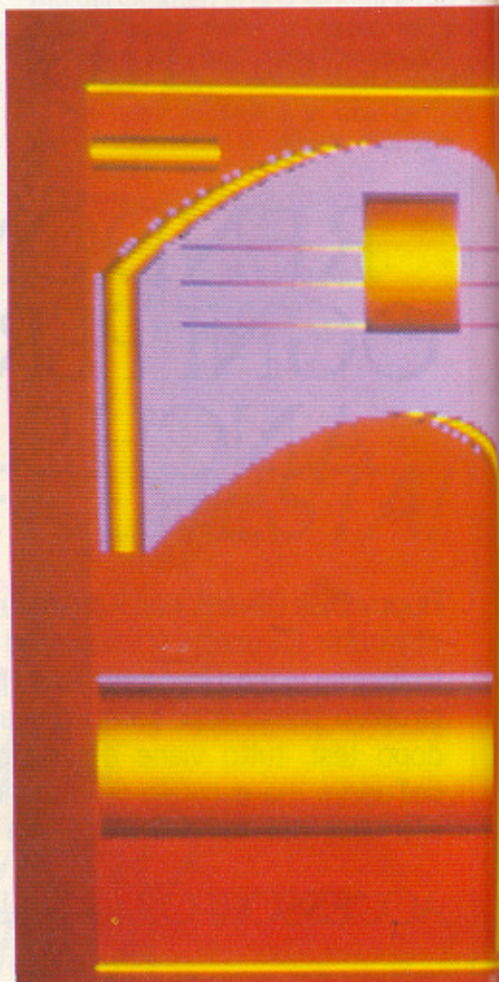
Una delle possibilità che hanno creato più aspettative attorno al PC 128S è senz'altro quella di poter manipolare file di tipo MS-DOS. Ovviamente, il computer della Olivetti Prodest non è un PC compatibile, come forse la sigla poteva erroneamente suggerire a qualcuno; infatti non è certamente possibile immettervi un dischetto contenente dei programmi sotto MS-DOS per poi farli girare.
Attualmente, sul mercato dei personal non esistono computer di questo tipo, così camaleontici. I problemi da risolvere sarebbero enormi: primo fra tutti la differenza delle CPU.

Ciò che invece è possibile fare, e che infatti la Olivetti Prodest ha pensato bene di mettere in pratica, è di passare dei file di tipo ASCII, anche se non puri, da un sistema all'altro, cioè dal MOS al MS-DOS.

Vista la grande diffusione delle macchine MS-DOS, specialmente in quegli ambienti di lavoro dove maggiormente è richiesto l'uso dei file sopra descritti, e vista, di pari passo, la grande differenza di costo, anche di "mantenimento", che intercorre tra un qualsiasi sistema MS-DOS ed un sistema PC 128S, è quasi scontato affermare che i motivi per considerare valido il programma "MS-DOS COPY FILES" sono più di uno.

Immaginate di avere in ufficio un PC compatibile, per mezzo del quale siete soliti scrivere degli articoli o altro, e, per i motivi più svariati, non ultimo l'ispirazione, immaginate di voler continuare il lavoro, iniziato in ufficio, a casa vostra. A questo punto, o siete in possesso di un altro PC compatibile, oppure potete utilizzare il PC 128S. Con quest'ultimo potrete ampliare, o comunque cambiare tranquillamente i testi da voi prodotti con il word processor dell'ufficio, sul vostro fido View, per poi riportarli nuovamente sul sistema MS-DOS. Sperando, questa è una delle poche cose che il PC 128S non sa ancora fare, che almeno l'ambiente familiare vi abbia ispirato.

MS-DOS copy file

Il programma "MS-DOS COPY FILE", una volta lanciato, ci presenta un menu composto da 7 voci, posto al centro dello schermo. Queste sono selezionabili o per mezzo dei testi cursore, oppure dalla pressione del numero corrispondente alla loro posizione nella lista.

1. TRANSFER FILE:

È la prima opzione disponibile e porta, a sua volta, ad un altro menu:

1.1 Copy Files from MS-DOS to PC 128S

1.2 Copy Files from PC 128S to MS-DOS

1.3 Main Menu

L'opzione 1.1, svolge la funzione di trasferire i file da un dischetto formattato in MS-DOS ad un dischetto utilizzabile su PC 128S. Alla sua attivazione, il programma provvederà a stampare sullo schermo la directory del dischetto, MS-DOS, posto nel drive attualmente selezionato; in seguito verrà riportata una linea di input caratterizzata dalla presenza di un prompt:
Files (S)
nella quale dovrà essere riportato il nome del file, contenuto nella directory, che vogliamo traferire. Tale filename deve comprendere anche la sua estensione. Dopo la pressione del testo Return, il programma provvederà a trasferire il file MS-DOS nella direcotry e nel drive MOS precedentemente selezionati.

Nel caso in cui il file richiesto non venisse trovato, il computer emetterà un messaggio del tipo:

File not found.

Se invece l'operazione giunge a termine, il file salvato sulla directory ADFS avrà, come filename, l'unione del nome originale alla propria estensione. È anche interessante notare che in questo procedimento, in appoggio ai filename, possiamo utilizzare dei caratteri wildcard per trasferire file multipli, o delle directory complete: #, *.

Esempi:

PIPPOFILE. TXT

Trasferirà il file MS-DOS PIPPOFILE TXT in una directory ADFS, sotto il nome di PIPPOFILETXT.
PIPPO *.*

Trasferirà tutti i file MS-DOS i cui nomi iniziano con PIPPO, e aventi una qualsiasi estensione, in una directory ADFS.

.

Traferirà tutti i file MS-DOS contenuti nella corrente directory, in una directory ADFS.

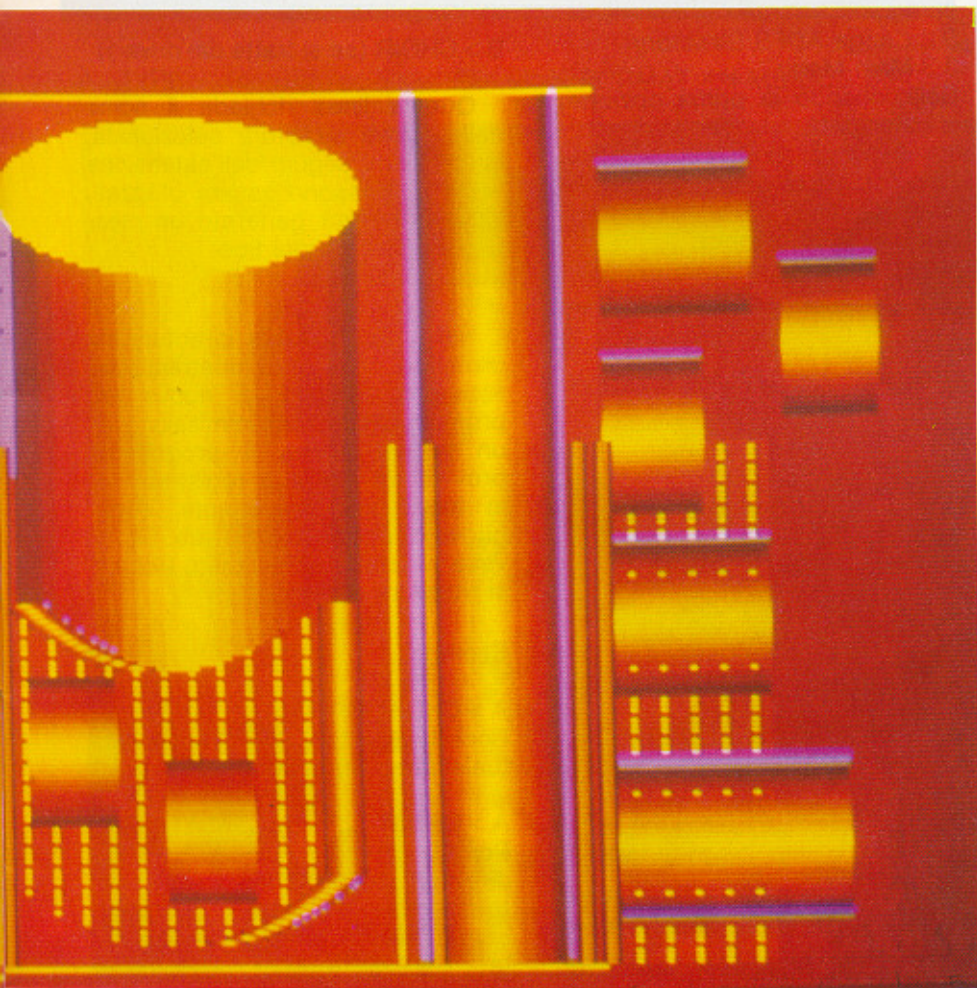
L'opzione 1.2, compie l'operazione inversa da quella vista, cioè trasferisce dei file di tipo ADFS su directory sotto MS-DOS.

All'attivazione del comando, il programma visualizza la directory ADFS, contenuta sul dischetto posto nel drive selezionato. Come visto in precedenza, anche in questo caso è presente una linea di input richiedente il nomefile da convertire.

Ci sono alcune differenze minime tra l'esempio attuale e il precedente: il nome del file trasferito conterrà solo le prime otto lettere del filename originale, mentre le rimanenti andranno a formare l'estensione.

Anche in questo caso, al fine di facilitare particolari immissioni di dati, si possono utilizzare i caratteri wildcard.

L'opzione Main Menu è presente in tutti i sotto menu e serve a punto per ritornare al menu principale.

2. DISPLAY DIRECTORY:

La seconda opzione, utile per poter leggere il contenuto delle directory sia dei dischetti sotto MS-DOS che sotto ADFS, riporta ad un sotto menu così composto:

2.1 Display MS-DOS Directory

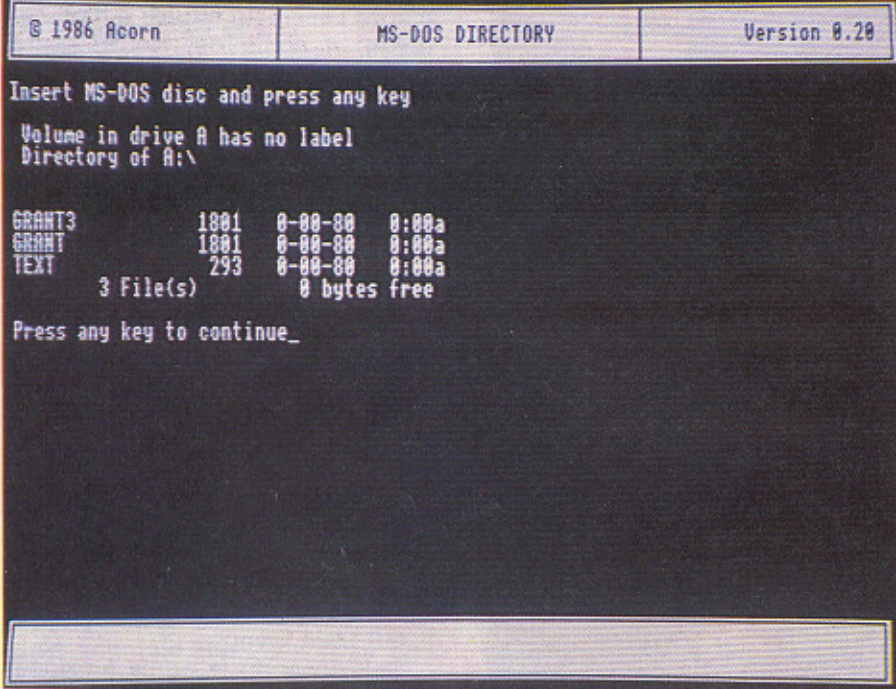
2.2 Display ADFS Directory

2.3 Main Menu

Selezionando la prima funzione,

sul monitor apparirà la seguente richiesta di input:
Wide Display (Y/N)?

Questa chiede se la corrente directory del dischetto sotto MS-DOS deve essere mostrata in modo esteso o un formato normale. Il formato esteso mosterà i filename associati alle loro estensioni disposti sullo schermo in file di cinque, senza che siano mostrati i parametri corrispondenti alla loro data e ora d'immissione o alla loro dimensione.

Al contrario, il formato normale mosterà i filename incolonnati uno ad uno e seguiti dai rispettivi parametri di estensione: lunghezza, tempo e data.

Alla fine della directory viene mostrato il numero di file in essa contenuti e lo spazio libero, misurato in byte.

L'opzione 2.2. mosterà la directory del dischetto , sotto ADFS, posto nel drive attualmente selezionato.

3. CHANGE DIRECTORY:

Questa opzione permette di cambiare il nome directory selezionata, contenuta nel drive corrente: sia sotto MS-DOS che in ADFS.

3.1. Cange MS-DOS Directory

```
© 1986 Acorn        MS-DOS DIRECTORY        Version 0.20

Insert MS-DOS disc and press any key

 Volume in drive A has no label
 Directory of A:\

GRANT3          1801   0-00-80   0:00a
GRANT           1801   0-00-80   0:00a
TEXT             293   0-00-80   0:00a
       3 File(s)          0 bytes free

Press any key to continue_
```

3.2 Cange ADFS Directory
3.3 Main Menu.

Selezionando la prima opzione, verrà stampata sul video la directory impostata correntemente del dischetto MS-DOS posto nel corrente drive. Come al solito, in coda apparirà una linea di input:

A>CHDIR (se si tratta del drive 0)
B>CHDIR (se si tratta del drive 1)

Questa linea richiede il nome della nuova directory selezionata. Solo i nomi seguiti dall'estensione <DIR> possono essere utilizzati, altrimenti verrà generato un messaggio d'errore del tipo:
Invalid directory

Nel caso in cui la directory sia presente, essa verrà mostrata sullo schermo, seguita di nuovo dal prompt CHDIR. All'immissione di un nuovo nome, il programma provvederà alla sua registrazione, in luogo del precedente, come nuova directory selezionata. Se invece si preme solo il tasto Return, si ritornerà al menu.

Esempi:

A>CHDIR PIPPO
Pone la directory PIPPO come l'attuale directory selezionata.

A>CHDIR /PIPPO/INSALATA
Pone la directory INSALATA, sotto-directory di PIPPO, a sua volta sotto-directory della Root directory, come l'attuale directory selezionata.

L'opzione 3.2 svolge la stessa

funzione su dischetti sotto ADFS. Anch'essa mosterà la Root directory seguita, però, da un altro prompt:

* Directory

L'uso di questo comando è identico a quello appena visto.

Per comprendere meglio l'uso di questo comando è opportuno effettuare una variazione di directory, per poi, ritornando al menu principale, selezionare l'opzione "Display Directory". A questo punto sullo schermo non comparirà più la Root directory, ma la directory appena impostata.

4. TYPE AN MS-DOS FILE:

La quarta opzione è molto utile, perché permette di riportare sul monitor il contenuto dei file sotto MS-DOS. Alla sua attivazione compare la directory principale seguita dal prompt:

A>TYPE

Dopo l'immissione del filename, seguito dalla sua estensione, sullo schermo verrà stampato il contenuto del file. È opportuno dire che in questo caso, diversamente da esempi visti in precedenza, i caratteri wildcard non sono utilizzabili.

Se il file richiamato non è presente, comparirà il messaggio d'errore: File not found.

Per fermare momentaneamente lo scorrimento del testo, basterà premere i tasti SHIFT e CTRL contemporaneamente; per continuare basta rilasciarli. L'uscita dall'opzione viene ottenuta con la pressione del tasto ESCAPE.

Per i file sotto ADFS esiste un'opzione simile, ma è selezionabile direttamente, per mezzo del comando ADFS "* TYPE". Per poter utilizzare i comandi ADFS, selezionare l'opzione "6. *Command" del menu principale.

5. FORMAT AN MS-DOS DISC

Come è palesemente indicato dal suo nome, questa opzione può essere utilizzata per formattare dischetti sotto MS-DOS, anche per mezzo del PC 128S.

Alla sua selezione, il programma mostrerà il prompt:

Single Side Drive (Y/N) ?

Premendo Y, si comunicherà al computer che si sta utilizzando un drive a faccia singola o comunque che si desidera formattare una sola parte di un dischetto. La scelta di N, ovviamente, provocherà la formattazione di tutti e due i lati del dischetto.

I dischetti così formattati risulteranno di 40 tracce di 9 settori ciascuna. Ciò è l'equivalente dell'MS-DOS 2. Quanto appena detto ha anche un altro significato: vuol dire che con questo programma non è possibile usare dischetti e quindi programmi creati sotto un'altra versione di MS-DOS.

```
>*CAT
$                   (22)
Drive:0             Option 03 (Exec)
Dir. $              Lib. "Unset"

!BOOT      WR (06)  APP        DLR(07)  DIRFILE    WR (11)  FUN        DLR(08)
MENU       WR (22)  MISC       DLR(09)  PMFS       WR (01)  SPRITE     WR (05)
SPRITER    WR (02)  WIMPSPR    WR (03)
>*TYPE DIRFILE
   Personalised Desktop
 PC 128S Olivetti Prodest
 G.& C.  Febbraio 1987
3

DIR Applications,APP
Wordprocessor View,VIEW
ViewSheet,VSH

DIR Sound & Vision,FUN
Sound1,SOUND1
Sound2,SOUND2
Sound3,SOUND3
Triangle,TRI
Square,SQ
Filled Triangle,FTRI
Filled Square,FSQ

DIR Miscellany,MISC
Return to BASIC,QUIT
Return to View,*WORD
Return to VSheet,*SHEET
>_
```

Dopo aver scelto il tipo di formattazione, apparirà il messaggio: Insert new diskette for drive a: and strike any key when ready eseguendo il quale si otterrà un nuovo dischetto formattato sotto MS-DOS, da utilizzare su qualsiasi computer PC compatibile.

Nel caso in cui si incorresse in un errore, verrà generato il relativo messaggio: Format failure, seguito da una richiesta di sostituzione del dischetto incriminato.

6.*COMMAND:

Questa opzione permette di immettere, in modo diretto, i comandi del MOS del PC 128S. È da qui che si potrà intervenire sulla gestione del computer riguardante L'ADFS, come già esposto in parte al punto 4. Un altro esempio dell'utilizzo dell'opzione 6. è il cambio del dischetto sotto ADFS. Ormai è a tutti noto che bisogna comunicare al sistema l'avvenuto cambiamento, per mezzo del comando *MOUNTn, dove nè il numero del drive selezionato ed è proprio da qui che ciò deve essere fatto.

L'ultimo comando utilizzabile è il "7.QUIT", che ci riporta alle normali condizioni d'utilizzo del PC 128S. Dopo questo breve viaggio, non ci resta molto da dire su questo programma, che riesce a sfruttare le potenzialità di base offerte dal PC 128S, se non di provarlo in tutte le sue parti, al fine di acquisire quei primi rudimenti, necessari all'esplorazione di quell'universo che va sotto il nome di MS-DOS.

ADFS

Facile gestione dei dischetti

2ª Parte

Dopo l'introduzione generale a quella parte del Sistema Operativo riguardante la gestione dei dischetti, effettuata lo scorso numero, in questo articolo cominceremo a prendere in considerazione un comando alla volta, cercando di mettere in risalto quelle che sono le specifiche particolarità di ognuno di questi. Visto il gran numero dei comandi relativi alla gestione dei drive, questa analisi verrà effettuata in due riprese. L'esposizione verrà svolta seguendo l'ordine alfabetico e sarà comprensiva della sintassi e della descrizione dei parametri:

*COMMANDO parametri
(Abbreviazione)

COMMENTO.

*ACCESS namefile [D;E;L;W;R]
(*AC.)

COMMENTO:
Imposta gli attributi di accesso per un file o un gruppo di file.

Gli attributi dei file vengono assegnati in accordo con la presenza, o meno, di uno o più, dei parametri seguenti:
L Protegge un file da cancellazioni o sovrascritture accidentali. Se un file protetto in questo modo, viene caricato in memoria, esso non potrà più essere salvato con lo stesso nome. Anche le directory possono venir protette per mezzo del comando L. C'è da tener presente che tale protezione è ininfluente in caso di riformattazione del dischetto.
W Imposta in "accesso in scrittura" un file. In tal modo è permesso portare delle variazioni al file interessato, da utilizzarsi solo con file.
R Imposta in "accesso in lettura" un file. In tal modo è possibile leggere e caricare in memoria un file; è utilizzabile solo con file.
E Imposta il file in "accesso solo per esecuzione" e rimuove eventuali comandi del tipo W e R. Questo parametro è utilizzato per proteggere programmi in codice macchina e, quando è attivato, impedisce l'uso del comando *LOAD e inibisce la possibilità di richiedere informazioni per mezzo di *EX e *INFO. I soli comandi utilizzabili sono *RUN, *<filename>, *DELETE, *REMOVE, *DESTROY.
D Questo attributo è utilizzato nel caso in cui l'oggetto sia una directory.

Gli attributi di default sono:
Files: W R
Directory: D L R

*BACK
COMMENTO:
Utilizzabile per rendere corrente la directory precedentemente selezionata.

In addizione alla corrente directory, l'ADFS mantiene in memoria un record delle ultime directory selezionate, le quali, per mezzo del comando *BACK, possono essere trasforamte nelle directory correnti.

*BACKUP <drive number> <drive number> (BAC.)

COMMENTO:
Legge tutte le informazioni contenute sulla superficie di un dischetto e le trasferisce su di un'altra: produce due dischetti uguali.

Questo comando copia l'intero contenuto del primo <drive number> al secondo <drive number>; se i due parametri sono uguali, l'ADFS considera che la copia dev'essere fatta nello stesso drive e che l'utilizzatore provvederà alla sostituzione alternata dei dischetti sorgente e destinazione, quando richiesto.

Tutte le eventuali informazioni contenute nel dischetto destinazione, verranno inesorabilmente perse.

C'è da precisare inoltre, che il contenuto della memoria verrà cancellato da questo comando: pertanto è consigliabile salvare l'eventuale programma in memoria, prima di utilizzarlo.

La velocità operativa del comando *BACKUP dipende dall'ammontare di memoria utilizzabile come buffer: le prestazioni migliori si ottengono sia usando il modo 7, che un qualsiasi modo Shadow.

*BUILD filename (*BU.)

COMMENTO:
Crea un file contenente linee susseguenti di input provenienti da tastiera.

Questo comando apre un nuovo file chiamato filename; tutte le successive linee di input provenienti dalla tastiera verranno direzionate nel file. Il file sarà considerato chiuso solo dopo la pressione del tasto ESCAPE.

Nel caso in cui sul dischetto esista un file dallo stesso nomefile di quello utilizzato in argomento a *BUILD, questo verrà perduto.

*BYE

COMMENTO:
Per chiudere permanentemente una sessione in ADFS.

Questo comando è simile, per effetto, al comando *CLOSE, che assicura la chiusura di qualsiasi file sequenziale aperto, con la scrittura dei relativi dati del buffer associato sul file.

*CAT <drive specification> <pathname>

Visualizza i nomi contenuti in un file catalogo. Sullo schermo apparirà il contenuto della directory/drive corrente, o specificata.

*CDIR <pathname> (*CD.)

Crea una nuova directory.
Il comando *CDIR crea una directory vuota, di nome specificato; nella posizione gerarchica indicata dal pathname.

*CLOSE (*CL.)

Chiude tutti i file sequenziali aperti.

Si assicura che tutti i file aperti siano chiusi e che i dati in RAM, non ancora scritti, vengano trasferiti nel file. Questo comando equivale al comando BASIC CLOSE #O.

*COMPACT <drive number> (*COM.)

Riorganizza la memoria libera dei dischetti, in modo da ottenere uno spazio vuoto continuo più ampio.

*COPY <drive number> <nomefile> <drive number>

Copia un file da un dischetto ad un altro o da una directory ad un'altra.

I contenuti di memoria possono venire sovrascritti da questo comando: sarà quindi opportuno salvarli prima di intraprendere una operazione di *COPY.

*CREATE <object> <dimensioni in byte> (*CR.)

Crea un file vuoto di una lunghezza specificata.

Il comando permette un semplice mezzo per riservare spazio a dei file su un dischetto. <dimensioni in byte> è un numero esadecimale indicante il numero di byte da riservare.

*DELETE <object> (*DE.)

Cancella un file dalla directory corrente, o una directory, se questa è vuota.

*DESTROY <object> (*DES.)

Cancella più file da un dischetto, con un unico comando.

Il comando *DESTROY è usato per cancellare un solo file, ma la possibilità di utilizzare i caratteri wildcard permette la cancellazione di più file contemporaneamente.

*DIR
<drive specification>
<directory name>

Cambia l'attuale directory corrente in una specificata in argomento.

*DISMOUNT
<drive number> (*DISM.)

Chiude tutti i file sequenziali correntemente aperti. Normalmente è da utilizzarsi prima di cambiare dischetto.

corrente (oppure specificata).

*EXEC<object> (*E.)

Accetta un imput da un file.
Il comando *EXEC legge i dati da un file dichiarato, byte per byte, come questo è stato costruito all'origine, tramite la tastiera.

*FORMAT
<drive number>
<dimensione> (*FO.)

Formatta il disco contenuto dal drive specificato con la dimensione

in argomento.

*FREE<drive number> (*FR.)

Mostra lo spazio libero utilizzabile sul dischetto.

*INFO<object>

Visualizza informazioni relative ad un file, a un gruppo di file se si utilizzano i caratteri wildcart.

*LCAT (*LC.)

Mostra il catalogo della libreria della direcotry corrente.
*LCAT equivale al comando *CAT con la libreria specificata nei

*DRIVE<drive number> (*DR.)

Cambia il drive corrente.
Il comando *DRIVE cambia il drive corrente in quello specificato.

*DUMP<object>
<start poin>
<offset> (*D.)

Mostra un Dump esadecimale e ASCII del file specificato.

*EX
<drive specification>
<pathname>

Mostra le informazioni relative a tutti i file contenuti nella directory

propri parametri.

*LEX

Mostra le informazioni relative ai file contenuti nella libreria.
*LEX fornisce esattamente le stesse facilitazioni di *EX, ma per le directory di libreria, senza avere la necessità di impostare la libreria a directory corrente.

*LIB<pathname>

Imposta la directory di libreria.

*LIST<object> (*L.)

Mostra un file contenente numeri di linea.
*LIST mostra il contenuto del file nominato in argomento, nel formato GSREAD, in cui:
— I codici ASCII da 32 a 26 vengono mandati al monitor, come i caratteri corrispondenti nel modo corrente.
— Tutti gli altri codici ASCII, eccettuato il 13, sono mostrati come espansioni del codice di controllo.
Ogni linea, per esempio una sequenza di codici ASCII chiusa dal comando RETUN, è preceduta da un numero di linea.

*LOAD<object>
<lad address> (*L.)

Carica in memoria un file specificato.

*MAP<drive number>

Mostra la mappa dello spazio libero, utilizzabile su un dischetto.
Mentre *FREE dà un'indicazione sull'ammontare totale della memoria libera, *MAP mostra una mappa della distribuzione della memoria libera, in termini di numero del settore di partenza ed estensione. Se si omette <drive number>, il drive considerato sarà quello corrente.
Come già accennato nell'introduzione, gli altri comandi mancanti verranno analizzati in un prossimo articolo: pertanto arrivederci al prossimo numero.

I DUE BASIC DEL PC 128

Proseguiamo l'elencazione delle parole chiave del BASIC residente nel PC 128. Come già visto nelle parti precedenti, i comandi più importanti sono accompagnati da brevi programmi esplicativi.

4° Parte

Per chi non avesse avuto modo di leggere gli articoli precedenti, ricordiamo che la struttura della spiegazione è omogenea per ogni comando e segue questa forma:

NOME DEL COMANDO

TIPO: Ovvero l'ambito d'uso del comando.
SINTASSI: Ovvero la forma corretta del comando, composta dalla parola chiave e dai suoi parametri.
COMMENTO: Ovvero la spiegazione del comando e dei suoi usi.

CHR$

TIPO: Funzione di conversione
SINTASSI: CHR$(n.)

COMMENTO:

Dove n. può essere sia un numero, che una variabile, che una funzione, il cui valore o risultato sia compreso tra 0 e 255.

Con questa funzione si ottengono i caratteri corrispondenti al valore ASCII riportato in argomento. Per la corrispondenza, vedere la tabella dei codici ASCII riportata in appendice del manuale.

Questo tipo di funzione è molto utile per introdurre nei programmi dei comandi particolari, come il ritorno di carrello, lo spostamento dei cursori, dei segnali acustici ed altri ancora.

I caratteri aventi un numero superiore a 127, sono caratteri definibili dall'utente per mezzo del comando DEFGR$. Ciò vuol dire che CHR$(128+n.) è equivalente a GR$(n.), cioè CHR$(128) è equivalente a GR$(0), CHR$(129) a GR$(1) e così via.

Altra caratteristica, che rende molto flessibile l'uso di questo comando è la possibilità di sommarlo, pertanto si potrà scrivere:

```
10 PRINT CHR$(12)
20 A$=CHR$(80)+CHR$(73)+
CHR$(80)+CHR$(80)+CHR$(79)
30 PRINT A$
```

```
10 PRINT CHR$(12)
20 A=(3*9)+(8*7)
30 PRINT CHR$(69)
40 PRINT CHR$(A)
50 PRINT CHR$(75-2)
```

```
10 PRINT CHR$(12)
20 FOR A=65 TO 90
30 PRINT CHR$(7);
40 PRINT CHR$(A)
50 FOR B=0 TO 200: NEXT B,A
```

CINT

TIPO: Funzione di conversione.

SINTASSI: CINT(n.)

COMMENTO:
Dove n. può essere una variabile, un numero o un'espressione.
Converte un numero di una precisione qualunque, in un numero intero, arrotondato al valore più vicino.
Se il range dei valori è esterno a −32768 e 32767, si avrà un errore di tipo Owerflow.

```
10 CLS
20 A=5.50
30 PRINT CINT(A)
40 PRINT CINT(5,49)
50 PRINT CINT(5.32*14.34378920)
```

CIRCLE

TIPO: Funzione

SINTASSI: CIRCLE(nc, nr) ro, rv, color; inizio, fine

COMMENTO:
Dove nc e nr sono il punto centrale individuato dalla colonna n e dalla riga n espressi in cifre. Il parametro opzionale ro (raggio orizzontale) determina il tracciamento di un'ellisse di raggio orizzontale n e raggio verticale (rv) n, i cui valori sono espressi in cifre. Nel caso in cui il parametro ro venga omesso, sarà tracciata una circonferenza di raggio rv. Il parametro color determina il colore usato per tracciare la circonferenza, se omesso, viene usato il colore dei caratteri. Gli ultimi due parametri, se presenti, determinano l'inizio e la fine della parte di circonferenza, o ellisse, da tracciare. Gli angoli vengono calcolati in radianti, mentre la direzione positiva è quella oraria. Il modo di tracciamento è determinato dal terzo parametro dell'istruzione CONSOLE.

```
10 CLS
20 CIRCLE(160, 100), 98,0
30 CIRCLE(110,60),18,4
40 CIRCLE(110,60),8,12
50 CIRCLE(200,60),18,4
60 CIRCLE(200,60),8,12
70 CIRCLE(160,100),21,1
80 CIRCLE(160,140)50,20,11;0,3.14
```

CIRCLEF

TIPO: Comando

SINTASSI: CIRCLEF(nc, nr)ro,rv,color;inizio,fine.

COMMENTO:
A differenza di CIRCLE, CIRCLEF traccia ellissi o circonferenze, o parti di esse, riempite. Il riempimento è ottenuto sia per mezzo di opportuni pattern, che tramite il colore in argomento.
A dimostrazione della equivalenza delle due sintassi, di CIRCLE e di CIRCLEF, provare il seguente programma:

```
10 CLS
20 CIRCLEF(160,100),98,0
30 CIRCLEF(110,60),18,4
40 CIRCLEF(110,60),8,12
50 CIRCLEF(200,60),18,4
60 CIRCLEF(200,60),8,12
70 CIRCLEF(160,100),21,1
80 CIRCLEF(160,140)50,20,11; 0,3.14
```

CLEAR

TIPO: Istruzione

SINTASSI: CLEARsp, ind1, nc, ind2

COMMENTO:
Dove sp è lo spazio, in caratteri, che vogliamo riservare in memoria alle stringhe: per default questo valore è pari a 300 caratteri.
Il secondo parametro, ind1, va espresso in esadecimale e corrisponde alla zona di memoria, all'interno dei banchi di memoria com-

mutabili tramite l'istruzione BANK, da noi selezionata. Tale numero deve essere inferiore a &HDFFF.

Il terzo parametro, nc, fissa il numero massimo di caratteri definiti dall'utente e ne riserva in memoria lo spazio necessario.

Il quarto parametro, ind2, permette di riservare una zona di memoria fuori dai banchi di memoria commutabili, cioè fra ind2+1 e &H9FFF.

I parametri impostati con il comando CLEAR, possono essere variati solo tramite un'altra istruzione CLEAR.

CLEAR2500 Fissa a 2500 caratteri lo spazio disponibile per la memorizzazione delle stringhe.
CLEAR,&H8FFF Riserva lo spazio in memoria a partire da &H9000 fino a &H9FFF nel banco di memoria corrente e riserva interamente la memoria dei banchi di ordine maggiore. La memoria a banchi parte da &H6000 fino a &H9FFF.
CLEAR,,,&H4FFF Riserva un'area di memoria esterna ai banchi di memoria commutabile, cioè fra &H2100 e &H5FFF, a partire da &H5000 fino a &H5FFF. La memoria non commutabile si trova più in basso rispetto ai banchi, i quali iniziano appunto da &H6000.
CLEAR,,20 Riserva, nei banchi di memoria, uno spazio sufficiente a contenere 20 caratteri definiti dall'utente.

CLOSE

TIPO: Istruzione

SINTASSI: CLOSE #n.canale, #n.canale

COMMENTO:
Dove #n.canale è il numero del canale che desideriamo chiudere.

Per mezzo di questa istruzione possiamo chiudere il o i canali aperti in precedenza dall'istruzione OPEN.

Se l'istruzione CLOSE viene utilizzata senza specificare l'argomento, essa chiuderà tutti i canali aperti.

```
CLOSE #2, #5
CLOSE
```

CLS

TIPO: Istruzione

SINTASSI: CLS

COMMENTO:
Cancella qualsiasi istruzione dallo schermo, comprese eventuali finestre create con il comando CONSOLE e pone il cursore in alto a sinistra.

Un risultato equivalente è ottenibile tramite l'uso di PRINT CHR$(12).

COLOR

TIPO: Istruzione.

SINTASSI: COLORcol.fo,col.be.,inv.

COMMENTO:
Dove col.fo. è il numero corrispondente al colore del primo piano, col.be il colore dello sfondo, e inv. (può essere solo 0 o 1), se presente determina un'inversione fra il colore di fondo e quello del primo piano.

Nessuno dei parametri è obbligatorio.

```
10 CLS
20 COLOR1,6
30 PRINT"CIAO";
40 COLOR7,6
50 PRINT"PIPPO";
60 COLOR0,6
70 PRINT"COME STAI?"
80 END
10 CLS
20 A$="CIAO PIPPO COME STAI?"
30 FOR I=1 TO 21
40 X$=MID$(A$,I,1)
50 C1=INT(RND*7)+1
60 C2=INT(RND*7)+1
70 IF C2=C1 THEN 60
80 COLOR C1,C2,1
90 PRINT X$;
100 NEXT I
110 END
```

COMMON

TIPO: Istruzione

SINTASSI: COMMON variabili

COMMENTO:
Dove variabili è un elenco di variabili.

Tramite questo comando si possono proteggere le variabili poste di seguito al comando, affinché siano utilizzabili dopo l'unione di due programmi, tramite l'istruzione CHAIN. È di fondamentale importanza invece sapere che tutte le variabili esterne al COMANDO COMMON vengono azzerate dall'operazione sopra descritta.

L'elenco delle variabili può essere composto anche da array con i relativi dimensionamenti.

Una precauzione da prendere è di dichiarare la lista di COMMON in testa al programma, onde evitare di perdere per strada delle variabili, per esempio a causa di una eventuale perdita della riga contenente l'istruzione, proprio durante una operazione di concatenamento.

I comandi che possono rendere vano il comando COMMON sono: CLEAR; RUN; NEW; LOAD.

```
COMMON A,C$,S(20)
```

CONSOLE

TIPO: Istruzione

SINTASSI: CONSOLEhr,1r,mg,vl,mod

COMMENTO:
Dove hr è un numero e indica la riga corrispondente al lato superiore della finestra visualizzata.

Dove 1r è un numero e indica la riga corrispondente al lato inferiore della finestra.

Il terzo parametro, mg, è un numero e determina il modo grafico con la seguente corrispondenza numerica:

0-1 Il segno del disegno cancella i caratteri a sé sottostanti.
3-2 Il segno del disegno non cancella i caratteri sottostanti.

4-5 Il segno del disegno appare invertito rispetto al primo piano precedente. (OR)

6-7 Il disegno appare solamente nei punti in cui si è usato il colore del primo piano. (AND)

Se il valore dei parametri sopra elencati è pari, il colore verrà modificato insieme al primo piano. Da notare che i valori dal 2 al 5 del terzo parametro, non possono essere usati in Basic 1.

Il quarto parametro, vl, determina la velocità e la modalità di scorrimento all'interno della finestra definita dall'utente:

0 Lo scorrimento avviene ad una velocità base, spostando il disegno verso la parte alta della finestra.

1 Lo scorrimento avviene ad una velocità lenta, spostando 1 disegno verso la parte alta della finestra.

2 Lo scorrimento avviene come se si scrivesse su una pagina, cioè appena questa è riempita, il cursore ritorna in alto a sinistra e ricomincia a scrivere sopra il testo precedente, senza però far scrollare la finestra.

Il quinto parametro, mod, determina i modi grafici:

0 Immette nel modo a 40 colonne con 16 colori.

1 Immette nel modo a 80 colonne con 2 colori, quindi il segno sarà ottenuto senza tener conto del colore imposto.

2 Immette nel modo in cui ogni punto dello schermo può assumere un colore indipendente dai punti vicini, selezionabile tra quattro colori diversi.

3 Come il precedente solo che i colori selezionabili, in questo modo, sono 16.

4 Attiva il modo di visualizzazione a pagine, dove la pagina 1 utilizza il colore 1 e scrive sulla pagina 2 tramite PRINT con il colore 2.

5 Attiva il modo di visualizzare a pagine, dove la pagina 2 utilizza il colore 2 e scrive sulla pagina 1 tramite PRINT con il colore 1.

6 Attiva il modo di visualizzazione a sovrapposizione di pagine, con la scrittura, tramite PRINT, sulla pagina 2 con il colore 2.

7 Attiva il modo di visualizzazione a sovrapposizione di pagine, con la scrittura tramite PRINT, sulla pagina 1 con il colore 1. La pagina 1 ha priorità rispetto alla 2. Da 8 a 11 si attiva il modo di tripla sovrapposizione, dove:

8 Primo piano.

9 Secondo piano.

10 Terzo piano.

11 Quarto piano.

Ogni piano utilizza come colore il colore corrispondente al proprio numero. Dal modo 4 al modo 11, il colore 0 è usato come colore di sfondo. In modo 3 non viene gestita la visualizzazione dei caratteri, così dal modo 8 al modo 11 non viene gestita né la grafica né il testo.

Nei modi dal 4 al 7 la selezione della pagina grafica viene ottenuta tramite il comando COLOR con l'opportuno parametro posto da 0 a 3.

La complessità e la potenza del comando CONSOLE suggerisce un suo ulteriore approfondimento, ma soprattutto, e ciò è ancora più importante, richiede da parte dell'utilizzatore una prova precisa del comando stesso, tramite il suo uso e per mezzo di una puntuale memorizzazione delle variazioni grafiche, causate dal variare del valore dei parametri in argomento.

Di seguito riportiamo un breve programma che potrà dare utili indicazioni e, a costo d'essere monotoni, insistiamo nel consigliare il cambiamento dei vari parametri.

Solo con molte prove si riesce a destreggiarsi bene all'interno di un linguaggio di programmazione, pertanto non abbiate paura di fare del "male" al vostro PC 128, provate.

```
10 CLS
20 A$="===========-
==================-
==================-
=" '40 =
30 B$="0000000000000000000-
0000000000000000000" '40 0
40 COLOR4,8,0
50 CONSOLE,,,0
60 LOCATE0,24
70 PRINT A$
80 FOR I=1 TO 70
90 IF    I=35    THEN    CONSO-
LE8,17,0,1:LOCATE0,17:CO-
LOR2,0:A$=B$
100 PRINT A$
110 NEXT I
120 END
```

```
10 CLS
20 A$=
"=================-
=================-
=============" '40 =
30 COLOR0,8
40 LOCATE0,20
50 CONSOLE8,20,,0 ' Cambiare lo
0 in 1 e in 2
60 FOR I=1 TO 70
70 PRINT A$
80 NEXT
90 END
```

CONT

TIPO: Comando

SINTASSI: CONT

COMMENTO:

CONT è un comando diretto e non è possibile usarlo da programma. La sua utilità è estrema in quei casi in cui è necessario interrompere un programma tramite le istruzioni CTRL-C, END o STOP. Infatti, in tal caso, il programma riprenderà dal punto esatto in cui è stato interrotto. Bisogna però fare attenzione a non cambiare nessuna linea del programma, altrimenti otterremmo un messaggio d'errore del tipo Can't Continue. Nel caso in cui l'interruzione sia avvenuta in corrispondenza di un'istruzione di INPUT, questa verrà ripetuta e sarà quindi d'obbligo una nostra risposta alla domanda d'immissione dei dati richiesti.

Una eccellente facilitazione data dal comando CONT è di poter interrompere il programma e di farlo continuare dopo averne alterati alcuni valori di variabili: sempre però in modo diretto.

Per verificare quando detto, provate a far funzionare il breve programma sotto riportato, il quale non fa altro che scrivere sullo schermo una serie di numeri da 1 a 1000: questi sono i valori della variabile A. Dopo aver lanciato il programma interrompetelo premendo, contemporaneamnete, i tasti CTRL e C. A questo punto il programma si fermerà visualizzando il numero di linea in cui è giunto e il Prompt OK. Ora, in modo diretto, scrivete A=1, seguito da RETURN e impartite il comando CONT. Appena il programma inizierà a stampare numeri sul monitor, vedrete che il primo numero stampato sarà 2 (cioè 1 + 1).

```
10 CLS
20 FOR A=1 TO 1000
30 PRINT A
40 NEXT
50 END
```